Algorytmy i Struktury Danych, 11. ćwiczenia

2021-12-15

Zadanie 9.1

Podaj przykład najmniejszej uniwersalnej rodziny funkcji haszujących z uniwersum $\{1, 2, 3, 4, 5\}$ w przestrzeń adresową $\{1, 2\}$ (w postaci tabelki z wartościami każdej z funkcji).

Rozwiązanie: Rozwiązanie 1 (10 funkcji)

funkcja	h(1)	h(2)	h(3)	h(4)	h(5)
h_1	1	1	2	2	2
h_2	1	2	1	2	2
h_3	1	2	2	1	2
h_4	1	2	2	2	1
h_5	2	1	1	2	2
h_6	2	1	2	1	2
h_7	2	1	2	2	1
h_8	2	2	1	1	2
h_9	2	2	1	2	1
h_{10}	2	2	2	1	1

Dla każdego x, y $(x \neq y)$ mamy 4 funkcje takie, że $h(x) = h(y)$ i 6 funkcji z $h(x) \neq h(y)$.

Rozwiązanie 2 (4 funkcje)

funkcja	h(1)	h(2)	h(3)	h(4)	h(5)
h_1	1	1	2	2	2
h_4	1	2	2	2	1
h_6	2	1	2	1	2
h_{10}	2	2	2	1	1

Dla każdego x, y $(x \neq y)$ mamy 2 funkcje takie, że $h(x) = h(y)$ i 2 funkcje z $h(x) \neq h(y)$. Więc dla losowego wyboru funkcji haszującej i dowolnego x, y $(x \neq y)$ mamy: $P(h(x) = h(y)) = \frac{1}{2}$.

Zadanie 9.2

W tym zadaniu należy udowodnić, że opisana poniżej rodzina funkcji haszujących jest rodziną uniwersalną. Niech m będzie liczbą pierwszą. Przyjmijmy, że klucze pochodzą z uniwersum $U = \{0, 1, \ldots, m-1\}^{r+1}$. Innymi słowy, każdy element U to krotka $x = \langle x_0, x_1, \ldots, x_r \rangle$, gdzie x_i jest liczbą ze zbioru $\{0, 1, \ldots, m-1\}$. Dla ustalonej krotki $a = \langle a_0, a_1, \ldots, a_r \rangle$, definiujemy funkcję

haszującą

$$h_a(x) = \sum_{i=0}^{r} a_i x_i \mod m$$

Udowodnij, że rodzina $H_m = \{h_a : a = \{0, 1, \ldots, m-1\}^{r+1}\}$ jest uniwersalną rodziną funkcji haszujących.

Wskazówka: rozważ dwa różne klucze x oraz y i bez straty ogólności załóż, że $x_0 \neq y_0$. Wykaż, że liczba tych a, dla których $h_a(x) = h_a(y)$ wynosi m^r. W tym celu pokaż, że dla każdego z m^r wyborów ciągu $\langle a_1, \ldots, a_r \rangle$ istnieje tylko jedno a_0, że $h_a(x) = h_a(y)$.

Zadanie 9.3

Przygotowanie tablicy haszowanej z zadania 9.2 wymaga wygenerowania liczby pierwszej m i zainicjowania tablicy o rozmiarze m. W tym zadaniu zaproponujemy algorytm znajdowania liczby pierwszej m działający w czasie $o(m)$. W tym celu skorzystamy z faktu, że dla każdej dodatniej liczby całkowitej k, w przedziale $[k^3, (k+1)^3]$ znajduje się co najmniej jedna liczba pierwsza. Zastosuj metodę Sita Eratostenesa i wykaż, że można ją zaimplementować w czasie $O(k^2 \ln k) = o(m)$.

Rozwiązanie:
Zauważmy, że $(k+1)^3 - k^3 = O(k^2)$. Dodatkowo $\sum_{i=2}^{j} \lceil \frac{n}{i} \rceil = O(n \log n + j)$.

Algorithm 1: PrimeSearch(k)

```
isPrime[k^3, ..., (k+1)^3] = True
foreach i ∈ 2, ..., sqrt((k+1)^3) do
    j = i⌈k^3/i⌉
    while j ≤ (k+1)^3 do
        isPrime[j] = False
        j = j + i
```

Zadanie 9.4

Wykaż, że rodzina $H_{p,m} = \{h_{a,b} : a \in \{1, 2, ..., p-1\}$ i $b \in \{0, 1, 2, ..., p-1\}\}$ jest uniwersalną rodziną funkcji haszujących.

Rozwiązanie: Dla ustalonego m i p (p liczba pierwsza, $m < p$) definujemy rodzinę funkcji $H_{p,m}$ (dla $a \in Z_p^*, b \in Z_p$):

$$h_{a,b}(x) = ((ax + b) \mod p) \mod m$$

(na podstawie Cormen, strona 234)

Niech $k, l \in Z_p$ ($k \neq l$). Rozważmy wartość funkcji na poziomie $\mod p$:

$r = (ak + b) \mod p$

$s = (al + b) \mod p$

Jeśli odejmiemy równania stronami:

$r - s \equiv a(k - l) \mod p$

Ponieważ $a \neq 0$ i $k - l \neq 0$ stąd ich iloczny musi być różny od zera ($\mod p$).

Czyli na poziomie $\mod p$ nie mamy kolizji.
Możemy nawet na podstawie par (r, s) i (k, l) jednoznacznie wyznaczyć funkcję która daje takie mapowanie par:

$a = (r - s)((k - l)^{-1} \mod p) \mod p$

$b = (r - ak) \mod p$

Teraz musimy oszacować jakie jest prawodpodobnieństwo, że dla losowych $r, s \in Z_p$ mamy $r \equiv s(\mod m)$.

$$\lceil p/m \rceil - 1 \leq (p + m - 1/m) - 1 = (p - 1)/m$$

Czyli prawodpodobnieństwo, że r i s ze sobą kolidują:

$$(p - 1)/m/(p - 1) = 1/m$$

Zadanie 9.5

Zaproponuj rozszerzenie algorytmu Floyda-Warshalla tak, żeby można było odzyskać w czasie O(n) najlżejszą ścieżkę pomiędzy dowolnymi wierzchołkami a, b.

Rozwiązanie:
Algorytm oblicza najkrótsze ścieżki pomiędzy każdą parą wierzchołków. Algorytm działa poprawnie, jeśli w grafie nie istnieje cykl u ujemnej wadze.

---

foreach $i, j \in V$ **do**
  $dist[i, j] = d[i, j]$
  $next[i, j] = j$
foreach $i \in V$ **do**
  foreach $v_1 \in V$ **do**
    foreach $v_2 \in V$ **do**
      if $dist[v_1, v_2] < dist[v_1, i] + dist[i, v_2]$ **then**
        $dist[v_1, v_2] = dist[v_1, i] + dist[i, v_2]$
        $next[v_1, v_2] = next[v_1, i]$
